CS212

Even More Pointers
Fall 2006

Announcements

- P4: due Friday with 2 day extension to Sunday (noon, maybe later)
- School evaluations: link on website
- Game Design Showcase: Tue 12/5, 3-6PM, 361 Upson

Pointer Reminders

- Pointers:
  - Operators:
    - `&v`: address of `v`
    - `*p`: pointer `p` and the `p`'s *pointee*
  - Pointers:
    - `*p = exp`: L-value (insert `exp` @ `p`'s pointee)
    - `blah(*p)`: R-value (retrieve value from `p`'s pointee)
- Memory:
  - `malloc(int)`:
    - allocate space on heap
    - return address (type `void*`)
    - cast with `<type*>`
  - `free(voidpointer)`:
    - must free allocated heap space

Pointer Arithmetic Reminder

- Connection between malloc, pointers, and arrays?
  - what does `malloc(int)` do?
  - what does `*(p+int)` do?
- Picture:

Longer Example (C)

```
char* append(char* s1,char* s2,int size);

int main(int argc, char* argv[]) {
  char* s1; char* s2; char* s;
  int L1; int L2;
  int i; int size;

  L1 = 3; L2 = 2; size=L1+L2;

  s1=(char*)malloc((L1+1)*sizeof(char));
  s2=(char*)malloc((L2+1)*sizeof(char));

  *(s1+0)='a';
  *(s1+1)='b';
  *(s1+2)='c';
  *(s1+3)='\0';
  *(s2+0)='d';
  *(s2+1)='e';
  *(s2+2)='\0';

  s=append(s1,s2, size);
  for (i = 0; *(s+i) != '\0'; i = i+1)
    printf("%c", *(s+i));

}
```

```
char* append(char* s1,char* s2,int size) {
  char* s;
  int i;
  int j;

  s = (char*)malloc((size+1)*sizeof(char));

  for (i = 0; *(s1+i) != '\0'; i = i+1)
    *(s+i) = *(s1+i);

  for (j = 0; *(s2+j) != '\0'; j = j+1)
    *(s+i+j) = *(s2+j);

  *(s+i+j) = '\0';

  return s;
}
```


Pointer to Pointers

- Grammar allows **`int** p;`**
- What does that mean?
  - **`p`** is a pointer to a pointer to an int
  - **`p`**→**`*p`**→**`int`**
  - also: **`(int *)*`**
- Example:

```
int i; int* q; int **p;
i = 10;
q = &i;
p = &q;
printf("%s%i\n","**p:  ",**p);
// output? picture?
```


More Syntax

- ANSI C:

```
int main(int argc, char* argv[]) {
   int** p;
   p = (int **)malloc(sizeof(int*));
   *p = (int *)malloc(sizeof(int));
   **p = 10;
   printf("%s%i\n","**p:  ",**p);
}
```

- Arrays and Pointers:
  - array: **`a[i][j][k]`**
  - pointers: **`*( *( *(a+i) + j ) + k )`**

Arrays in C

```
int main(int argc, char* argv[]) {

  int **p;
  p = (int **)malloc(2*sizeof(int*));
  *(p+0) = (int *)malloc(4*sizeof(int));
  *(p+1) = (int *)malloc(4*sizeof(int));

  // row 0:
  *(*(p+0) + 0) = 0;
  *(*(p+0) + 1) = 1;
  *(*(p+0) + 2) = 2;
  *(*(p+0) + 3) = 3;

  // row 1:
  *(*(p+1) + 0) = 10;
  *(*(p+1) + 1) = 11;
  *(*(p+1) + 2) = 12;
  *(*(p+1) + 3) = 13;

  int i, j;
  for ( i = 0 ; i < 2 ; i++ ) {
    for ( j = 0 ; j < 4 ; j++ )
      printf("%i%s",*(*(p+i)+j)," ");
    printf("\n");
  }

  free(*p);
  free(*(p+1));
  free(p);
  return 0;
}
```